CSC418/2504: Fall 2007 Assignment 2

Date handed out: Oct. 17, 2007.
Due date: W Nov. 7, 2007. (written part in class, prog. part electronically by midnight).

Written part (50 marks total)

1. *Transformations* [10 marks]

   A 2D affine transformation is completely specified by its effect on three non-colinear points, i.e., by how it maps a triangle into another triangle. Find the 2D affine transformation that maps points (2, 3), (1,2), and (3, -1) into points (5, 4), (4, -2) and (1, 0) respectively. *Hint:* Formulate the problem as a linear system of equations for the unknowns. You might find that you can formulate two sets of equations, each in three unknowns. Also, the question has been crafted to give you a solution comprising integers.

2. *Animation* [10 marks]

   Say you wish to animate a clock pendulum anchored at the origin with the pendulum weight hanging vertically down (-Y axis) in the XY plane. The pendulum swings through 2*k* radians from end to end, "**easing in & out**" of the rotation at the ends of a swing that takes *m* seconds. The pendulum is at one extreme, say *k* radians at *t=0*. Ease-in ease-out, a term used by animators to indicate that the motion at the start and end of given time interval has zero velocity can be captured for one swing from one extreme to the other by the function sin(a), $-\pi/2 <= a <= \pi/2$. Write the 4x4 rotation matrix that would give the transformation of the pendulum at any given time *t*, to animate the oscillating pendulum.

3. *Viewing and Projection* [20 marks]

   a. [5 marks] What is the function of a lens in a real camera, and how does the "focal length" and "aperture" of the camera lens affect the images created? (your description should be no longer than a few sentences).
   b. [15 marks] Let **p=(p_x, p_y, p_z)** and **q =(q_x, q_y, q_z),** be two endpoints of a line segment, defined in viewer or camera coordinates. Let **m = 0.5(p+q)** be the midpoint of the line segment. Defining perspective projection as in class and the textbook, with the optical axis along the negative Z-axis, let **p', q'** and **m'** be the perspective projections of **p, q** and **m** respectively. Use the mathematical form of perspective projection to determine whether **m' = 0.5 * (p' + q')**. If it is not true for all p and q, characterize the conditions under which it would be true.

4. *Change of Basis* [10 marks] Let there be two cameras, described in world coordinates by their respective frames of reference, (e1, u1, v1, w1) and (e2, u2, v2, w2), where e's are the eye points and u,v,w's the basis vectors.
   Show mathematically how, given a point p1 in the local camera centered frame of reference of camera 1, you would compute the local camera-centered coordinates of the point in the local camera-centered frame of reference of camera 2.

Programming part (50 marks total).

1. *Cartoon shading* [25 marks]
   In cartoon tracing, silhouette edges of a model are drawn as a black outline. For this question we define a silhouette edge to be one for which one adjacent face is a front face and the other a back face with respect to a given viewpoint. Cartoon tracing also outlines visible sharp features on an object. For this question we define a sharp feature edge as one for which the angle between its adjacent face normals is higher than some sharpness angle **s** (max value of 180 deg.). Write a program to display a cartoon shading of a mesh specified in a file <filename>.obj whose format is as follows:

```
v 0 0 0
v 1 0 0
v 0 1 0
v 0 0 1
f 3 2 1
f 1 2 4
f 3 1 4
f 2 3 4
e 5 0 0
g -1 0 0
t 0 1 0
s 85
```

here lines beginning with **v** represent a vertex followed by its location in 3D world space, lines beginning with **f** represent faces (triangles) where the numbers are an index to the vertices of the polygon (starting from 1), in

counterclockwise order, **e** represents the viewpoint, **g** the gaze direction and **t** the up vector. Use your discretion to choose appropriate values for field of view, near/far clip planes and aspect ratio,

for the camera to view the example object described above.

In general the file contains a number of vertices (one per line) written as **v <x> <y> <z>**, followed by a number of faces (one per line) written as

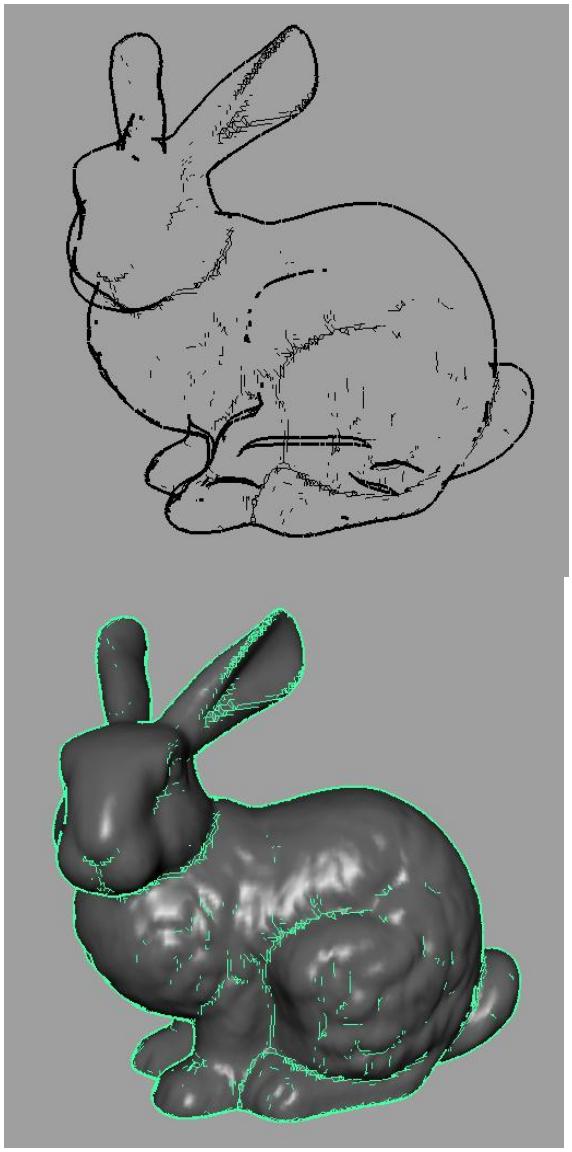
f <index1> <index2> ... <index n>. This is optionally followed by a viewpoint **e <x> <y> <z>**, view direction **g <x> <y> <z>,** up vector **t <x> <y> <z>** and sharpness angle **s <angle>**.

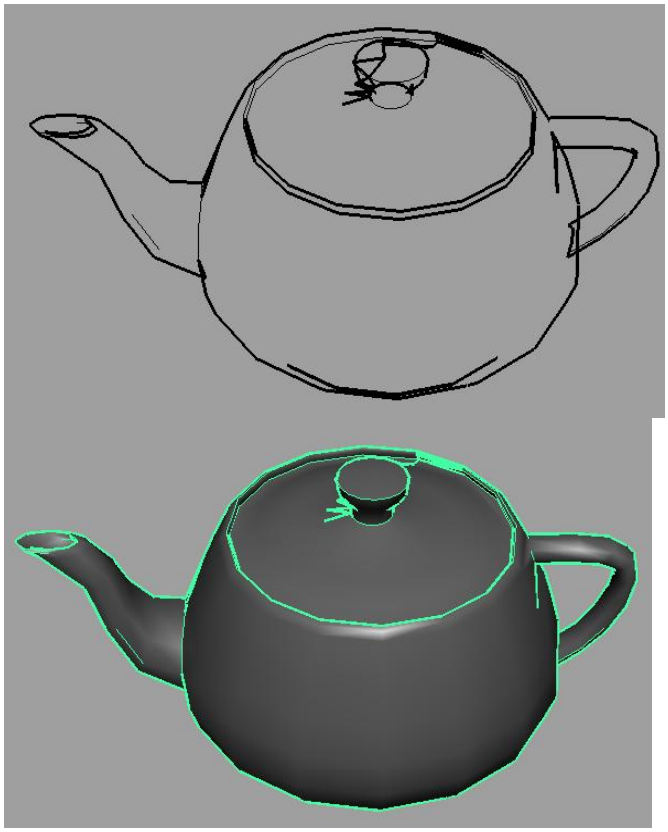
In the above, file import and set up of the basic view is worth 3 marks. 2 marks for setting up the faces, vertices edges in a structure, so you know what faces/vertices are adjacent to each other. 3 marks for drawing the correct cartoon edges.

Here are a couple of sample mesh files (this is the wavefront obj format. you can find any number of such files on the web.) These files do not have **e,g,t** and **s** values defined, pick these values for the models below as you see fit.

Bunny, Horse, Teapot

Sample renderings of the bunny and teapot are shown below. The thicker lines are silhouettes and the thin ones are sharp features.





[10 marks] Interactively *pan* the viewpoint based on left mouse click and drag. (*pan* translates the eye point of the camera along the up vector **t** corresponding to vertical motion of the mouse cursor on screen and along the vector given by the direction of **g X t** for horizontal motion of the mouse cursor on screen). *dolly* the viewpoint in and out along the view direction based on right mouse click and drag (dolly translates the eye point along the vector **g** corresponding to the horizontal motion of the mouse cursor on screen).

[15 marks] We now wish to shade the object by defining a color value at each vertex. The color is computed as a product of two colors **c1** and **c2** as **(c1.r*c2.r, c1.g*c2.g, c1.b*c2.b). c1** is based on the angle **a** which the vertex normal makes with the view direction **g**. **c1=(1,1,1)** or *white* for **a>=w (w=150^{o}** by default), **c1=(0,0,0)** or *black* for **a<=90^{o}** and a linearly interpolated grey for **90 ^{o} <=a<=w**. **c2** is based on the distance **d=||v-e||** of the vertex **v** from the camera viewpoint **e. c2=(1,0,0)** or *red* for **d<=r (r=5** by default), **c2=(1,0,0)** or *blue* for **d>=100** and a linearly interpolated color for **r<=d<=100**. Animate the shading with the click and drag of the middle mouse button, angle **w** maps to the horizontal motion of the mouse cursor and distance **r** maps to the vertical motion of the mouse cursor.

Submission:

Hand in your written work for the first part.

For the programming part, submit your code electronically along with a makefile and all files necessary for the program to compile.
Also submit a short readme.txt file explaining your program structure and any assumptions made, special considerations etc.

- `submit -N a2b csc418h file1 file2 ...`
- `submit -N a2b csc2504h  file1 file2 ...`